Proceedings of the 1986 Winter Simulation Conference
J. Wilson, J. Henriksen, S. Roberts (eds.)

SIMULATING WITH GPSS/PC

Ricardo F. Garzia
AT&T Bell Laboratories
Columbus, Ohio 43213

1. INTRODUCTION

In the study of systems one is often led to modeling and simulation as the most viable approach for investigating their properties and behavior. Following Schmidt and Taylor, we define a system as a collection of entities which act and interact together towards the accomplishment of some logical end. The system can be either naturally occurring, man made, or abstract. The importance of modeling and simulation in the study of such systems often results from the

- impossibility of dealing directly with the system (e.g., the system may not yet exist).
- cost of studying the system directly may be too high.
- impracticality of dealing directly with the system (e.g., time consuming).

Modeling the system consists of abstracting its features and properties paying particular attention to those which are of interest to the study. This is usually not just a reasonable approach, but in fact, a necessary approach in order to obtain a manageable model. That is, a model that can be properly analyzed to yield answers to the questions posed in the study. In fact, it is usually the case that all of the details concerning the system are not fully known making this the only possible approach.

The term *Simulation* has been defined in various ways. For the purposes of this tutorial, simulation is (loosely) considered to be the realization of a model, used to represent a system, in a form suitable for deriving information about the system's characteristics/properties of interest. By a *suitable form* we usually mean that we are able to compute (derive) qualitative and/or quantitative information which elucidates certain aspects of the nature of our system, usually taking the form of a computer program. Based on these ideas, it is clear that we must first have a model of the system before we develop the simulation. The system model itself can range from a basic conceptual model to a precise set of mathematical relationships describing the system's behavior. Based on the chosen representation, the associated model can be deterministic or stochastic in nature. Within each of these subdivisions we can further categorize the model in terms of the way in which its state changes as a function of time. The associated simulation will thus be one of three types: *continuous time, discrete time, or discrete event.*

1.1 Continuous Simulation-

Continuous time simulations arise from models described by ordinary differential or partial differential equations.

1.2 Discrete Time Simulation-

Discrete time simulations arise from models which can be expressed in the form of difference equations.

1.3 Discrete Event Simulation-

For this type of simulation, time evolves in discrete, nonuniform steps from one event to the next. The actual time at which these events occur can be any real number. Such simulations arise from systems which can be modeled as a sequence of countable events, where it is assumed that there is nothing of interest taking place between events. Examples of such systems include flexible manufacturing systems, queueing networks, communications networks, etc.

2. GPSS/PC ™-

Within the area of discrete event simulation languages developed to date, none has had more impact than GPSS, the **General Purpose Simulation System,** which was developed by Geoffrey Gordon of IBM Corporation around 1961.

GPSS/PC ™ is primarily an interactive simulation environment which supports a dialect of the GPSS simulation language. GPSS/PC includes about 95 percent of the block statements of GPSS V, the IBM mainframe version, and has more built-in state variables (System Numeric Attributes) than its mainframe cousin.

The execution time of GPSS/PC is satisfactory for most applications, running at about 1000 block entries per second on a 8 MHz IBM PC/AT. Similarly, with 640K of RAM memory, models with thousands of blocks and transactions (at any instant) can be run.

Recent developments have added 5 dynamic interactive graphics windows, and an animation window which can be coupled to GPSS simulations. To interact through the graphics windows, one can use a mouse, light pen, or the keypad as a pointing device. In addition, a HELP block allows access to FORTRAN routines and provides for continuous/mixed simulations, full user dialogues, real time data retrieval, real time process control, interface to other packages, and many other uses.

GPSS/PC is unique in its level of interactivity. A full complement of debugging commands allows the user to place stops, step through the simulation, and make modifications in the middle of a simulation. In addition, GPSS/PC allows GPSS block statements to be used as interactive commands. Essentially all simulation primitives are accessible to the user interactively. The motivation for this extremely high level of control is the philosophy that design alternatives, under scrutiny in the simulation project, are best conceived and modified when the simulation analyst has an accurate intuitive **feel** for the behavior of the system. This comes from closely interacting with the simulation environment, and is intended to be used only to prune the list of final design alternatives under study. The final simulations under which statistical significance is measured, must, of course, be run in an unperturbed environment. GPSS/PC now includes an ANOVA command to calculate confidence intervals and perform a one-way analysis of variance. GPSS/PC does not admit data into the results data base from simulations which were perturbed by interactions.

The interactive graphics windows allow users of GPSS/PC to observe and modify the dynamics of a running simulation visually. Each of the major GPSS entity types has a graphics window for observation and manipulation, possibly by a pointing device. In line with the GPSS/PC design objective of minimal hardware requirements, these features run on even monochrome IBM PC compatibles. However, an EGA (Enhanced Graphics Adaptor) board or compatible is needed for GPSS block symbols and user definable shapes. In addition to the major graphics windows, the user may open up to 4 small windows, called Microwindows ™. Each can contain a short title and a state variable, which is updated on the screen as the simulation runs.

Model development under GPSS/PC occurs in a single phase, rather than in a lengthy cycle of edits, recompilations, links, and executions. Thus, changes to existing models can be effected immediately without spending minutes of user time changing phases. In addition to the time saved, the simulation analyst does not lose the mental context under which the change was made. The net effect is that model development is generally much faster than in compiler environments. Similarly, GPSS/PC incorporates a grammar directed parser which is coupled to the keyboard. This makes it impossible for the user to make syntax errors from the keyboard. A new **swap-out** function allows users to suspend a simulation and do other, perhaps unrelated, work. These, and other features of GPSS/PC, are intended to speed up the simulation development time by catering to the user.

3. HOW GPSS/PC WORKS

To explore how GPSS/PC ™ works, we are going to follow two different approaches, which are:

- Becoming familiar with GPSS/PC ™
- Running a Simulation

3.1 Becoming Familiar with GPSS/PC ™

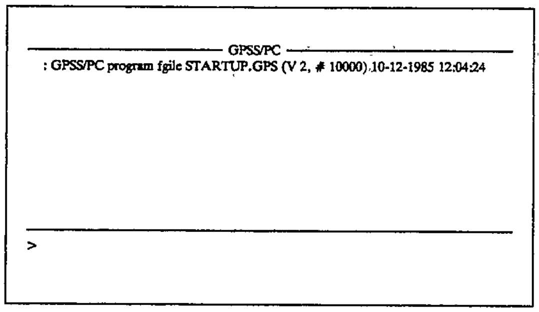
We turn on the PC and boot the MS-DOS operating system. After a few moments, the PC prompt appears as

```
C>    or    A>
```

The first prompt appears if we have a hard disk, the second one if we have a floppy disk. In any case, to start GPSS/PC ™ we just type

```
GPSSPC    [CR]
```

where [CR] is to be interpreted as carriage return, or the enter key. For some seconds we see the prologue screen which indicates the copyright statements as well as version number information. Then automatically, the data window appears on the screen. The data window is shown in Figure 1.



```
———————————— GPSS/PC ————————————
: GPSS/PC program fgile STARTUP.GPS (V 2, # 10000).10-12-1985 12:04:24

——————————————————————————————————
>
```

Figure 1- Data Window

To retrieve a GPSS model from a file, we just type

```
@SAMPLE1.GPS    [CR]
```

When you hear the audio signal GPSS/PC ™ is ready. In fact this is called the *ready sound*. Line numbered statements read from a file or entered through the keyboard are inserted into a data structure called the **Savable Program**. This is the ASCII representation of each line-numbered statement scanned by GPSS/PC ™. As its name implies, it can be **saved** in one or more DOS files, called program files.

The savable program can be seen on the screen, at any moment, using the display command

```
DISPLAY [CR]
```

An important fact to remember is that only statements with line numbers are put into the savable program. In fact, you use line numbers to tell GPSS/PC ™ where to insert your statement in the savable program. What happens when you want to insert an statement between statements 23 and 24. Very simple, just use a number with decimal fraction such as

```
23.6 BUFFER
```

The result is that the statement BUFFER will be inserted between statements 23 and 24.

Let us assume that we want to save the savable program in a file called TEST.GPS. Then we type

```
SAVE TEST.GPS    [CR]
```

Any GPSS/PC window can be selected by a single keystroke. For example, to open the Blocks Window, just press

```
[Alt+B]
```

The above entry is formed pushing together the keys [Alt] and [B]. This command allows the transfer from the Data Window to the Block Window, or

Data Window → Block Window

To start the simulation we type

```
START 300,NP    [CR]
```

where NP stands for no print, and 300 is the number of transactions to be terminated. NP suppressed the creation of the standard report.

3.2 Running a Simulation

We turn on the PC and boot the MS-DOS operating system. After a few moments the PC will prompt us that it is ready. To start GPSS/PC ™ we type

```
GPSSPC    [CR]
```

Now we would like to introduce the information needed to draw a plot, which is

```
PLOT Q$BARBER,200,0,28000    Barber Shop Waiting Line    [CR]
```

The plot command constructs plot axes and prepares the correct model to fill in the plot as the next simulation proceeds. The plot operands are as follows:

- A operand- Plot argument required. The operand must be SNA
- B operand- Maximum y value
- C operand- Start time of plot
- D operand- End time of plot

Then we start the simulation with the command

```
START 1000    [CR]
```

We can interrupt the simulation whenever necessary by using [home] or [Esc] any time needed. To continue the simulation we type

```
CONT    [CR]
```

Some of the SNA available in GPSS/PC are indicated in Table 1, which can be use in the operand fields and expressions of commands and statements. In al cases *entnum* must be replaced by any entity specifier. The entity specifier could be a name (preceded by a [$] separator) or a number, or for indirect addressing, it could be an asterisk, [*], followed by a name or number. There are a total of 52 SNA available in GPSS/PC.

TABLE 1. SNA

SNA	Comment
AC1	- Value of absolute system clock
C1	- Value of relative system clock
Q*entnum*	- Current count content
QA*entnum*	- Average queue content
QC*entnum*	- Total queue entries
QM*entnum*	- Maximum queue contents
QT*entnum*	- Average queue residence time
QZ*entnum*	- Queue zero entry count
S*entnum*	- Storage in use
SA*entnum*	- Average storage in use
TG1	- Remaining termination count
XN1	- Active transaction number
Z1	- Free memory

At this time, let's create some microwindows, such as

```
MI      1,AC1           ;Clock
MI      2,XN1           ;Active
MI      3,Q$BARBER      ;Queue
```

Let's turn to the Block Window, using the command [Alt+B]. The representation of the program in GPSS blocks will be seen on the screen, and on the right, the small microwindows just created. We have the capability to create up to four microwindows.

The graphic output available on GPSS/PC ™ is indicated in Table 1.

TABLE 2. Graphic Output

Command	Graphic Output
Alt+B	Block Window
Alt+F	Facilities Window
Alt+L	Trace Lines
Alt+T	Tables Window
Alt+M	Matrices Window
Alt+S	Storage Window
Alt+P	Position Window

4. *GPSS BLOCKS*

The statements in a GPSS program are either block statements, which cause a block to be created, or control statements which do not. If a block statement does not have a line number, it is applied to the active transaction without causing a block to be created in the current model. This is called **Manual Simulation.**

Block statements with line numbers are inserted into the savable program and the associated block is inserted into the corresponding position in the current model.

The development of a model which represents a network can be accomplished using the blocks and statements of the GPSS language.

The system to be simulated in GPSS is described as a block diagram in which blocks represent the activities, and lines joining the blocks indicate the sequence in which the activities are executed. There are exceptions to this flow of information as we will discuss later.

Before discussing each GPSS block, let us define the general structure of a block, which is composed of:

1. **Label**- Columns 2-6; can be omitted most of the time.
2. **Block Name**- Columns 8-18.
3. **Block Operands**- Columns 19-34; only the appropriate ones will be used.
4. **Comments**- Columns 35-80.

The above columns designation has been the classical approach in GPSS. GPSS/PC has free form input with automated spacing.

Therefore, the general structure of the GPSS block definition is

Label Block Name A,B,C,D,E Comments

As we see, each block description has several operands, starting with operand A, then operand B, and so on. These operands control the activity performed by the block.

Looking at Figure 2, we find twelve blocks, with their corresponding symbolic representation, to be discussed in our presentation. These are the most frequently used blocks. In general, GPSS/PC ™ has 48 blocks.

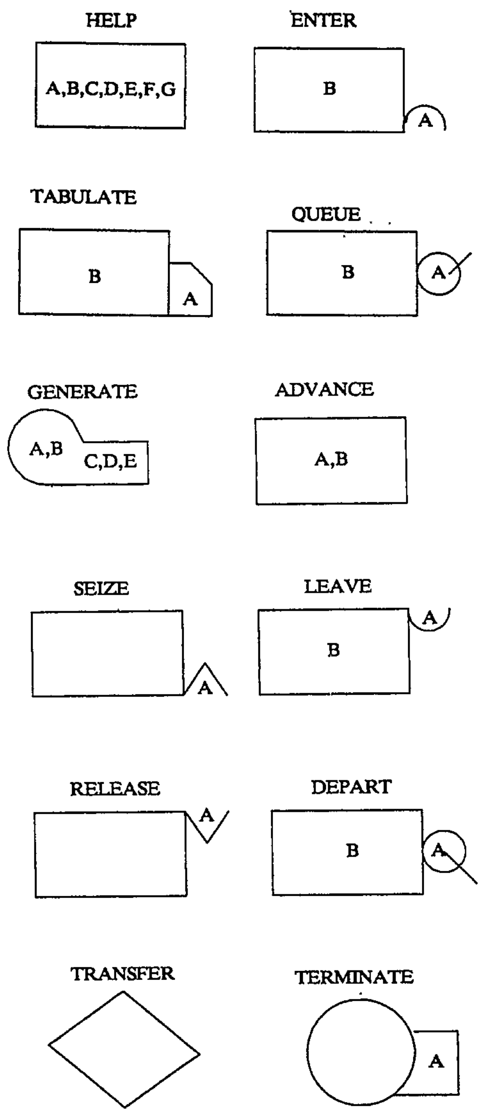


Figure 2- GPSS Bocks Symbols

4.1 GENERATE

A GENERATE block creates transactions for future entry into the simulation. The operands of the GENERATE block are

GENERATE A,B,C,D,E

- **Operand A-** Mean intergeneration time. This operand must be *null, name, posinteger*, or *direct*SNA. No transaction parameters can be used.
- **Operand B-** Intergeneration time half-range or function modifier. Optional. This operand must be *null, name, posinteger*, or *direct*SNA. No transaction parameter can be used.
- **Operand C-** Start delay time. Time increment for the first transaction. Optional. This operand must be *null, name, posinteger*, or *direct*SNA. No transaction parameter can be used.
- **Operand D-** Creation Limit. Optional, the default is not limit. This operand must be *null, name, posinteger*, or *direct*SNA. No transaction parameter can be used.
- **Operand E-** Priority level. Optional, the default value is zero. This operand must be *null, name, posinteger, or direct*SNA. No transaction parameter can be used.

Let us assume that the value specified by the A operand is α, and the one specified by B is β, then the generation of transactions occur at the times specified by

$$\alpha \pm \beta$$

which produces the generation of transactions at one of the times

$$\alpha-\beta, \alpha-\beta+1, \alpha-\beta+2, \ldots, \alpha, \alpha+1, \ldots, \textit{ or } \alpha+\beta$$

with equal probability given by

$$\textit{Probability} = \frac{1}{2\beta+1}$$

4.2 HELP

A HELP block passes up to seven savevalues to and from a FORTRAN program. The operands of the HELP block are:

HELP A,B,C,D,E,F,G

All these operands are the savevalue name or number to be used by the FORTRAN program. Optional. The operands must be *null, name, posinteger*, or SNA, or SNA *parameter.*

When a transaction enters a HELP block, any savevalue specified in operands A through G of the block statement is passed to the COMMON area of a previously linked FORTRAN subroutine. The subroutine is executed, and the (possibly changed) values are returned to the GPSS savevalues. If the FORTRAN COMMON variable named REPAINT is nonzero when the FORTRAN program finishes, GPSS/PC ™ repaints the computer screen. You can control the REPAINT value through FORTRAN statements in your HELP program.

In order to use HELP feature, you must combine a FORTRAN program with two other modules. This must be done before starting a GPSS/PC ™ session. There are three steps, which are:

1. Create the FORTRAN program, using a text editor.
2. The FORTRAN program need to be compiled, so that we obtain an object module.
3. You need to link the compiled program with two object modules provided in the GPSS/PC ™ distribution diskette. The resulting program, named GPSSMAIN.EXE, will start the GPSS/PC ™ session with the HELP program in place.

The FORTRAN program can be of any size, however it must have only one subroutine named HELP.

4.3 QUEUE

The QUEUE block never refuses entry to a transaction because it has no practical capacity limit. It is unit rather than transactions which are added to the queue. Transactions which enter a queue block are not delayed as the result of entering a QUEUE block; they attempt during the same clock time to enter the next sequential block. The operands associated with this block are: A, the queue name or number to which units are to be added; and B, the number of units to be added to the queue. The default value for this operand is one.

Because a queue is generally used to measure the amount of waiting time for an equipment entity, a QUEUE block is usually followed by a block such as SEIZE or ENTER, which can refuse entry to a transaction. QUEUE block is usually used in conjunction with the DEPART block.

4.4 DEPART

The DEPART block reduces the contents of a queue and updates usage statistics. The operands are: A, name or number of the queue from which units are to be subtracted; and B, number of units to be subtracted from the queue. This value must not exceed the number of units currently in the queue.

4.5 SEIZE

The SEIZE block will hold a facility until the transaction enters a RELEASE block, usually following completion of the service in the corresponding ADVANCE block. The A operand is the name or number of the facility.

4.6 RELEASE

The RELEASE block frees the facility that has been held by a transaction entering the SEIZE block. The A operand is the name or number of the facility to be released.

4.7 ADVANCE

An ADVANCE block delays the progress of a transaction for a specified amount of simulated time. The operands of the ADVANCE block are:

ADVANCE A,B

4.8 TERMINATE

A TERMINATE block removes the active transaction from the simulation and optionally reduces the termination count. The operand of the TERMINATE block is

TERMINATE A

Operand A- Termination count decrement. Optional, the default value is zero. This operand must be *null, name, posinteger*, or *direct*SNA.

4.9 *TRANSFER*

The *TRANSFER* block provides the means of diverting transactions to a nonsequential next block in the model. The following transfer modes are permitted:

TABLE 3. TRANSFER MODES

Mode	Condition
1	Unconditional
2	Fractional
3	Both
4	All
5	Pick
6	Function (FN)
7	Parameter (PX)
8	Subroutine (SBR)
9	Simultaneous (SIM)

The operands associated with this block are: A, transfer mode selected; B and C Operands- Possible next block values.

We consider only the mode that we are going to use in our application, which is the fractional mode. This mode specifies a three-digit number which indicates the proportion (in parts per thousand) of entering transactions that are to be diverted to the C operand next block. The selected exit is the only one tried by that transaction.

4.10 *ENTER*

A transaction passing through the ENTER block will hold a unit in the storage entity. The A operand is the name or number of the storage.

4.11 *LEAVE*

A transaction entering the LEAVE block frees one unit in the storage entity. The A operand is the name or number of the storage declaration.

4.12 *STORAGE*

The STORAGE block defines the capacity of a multi owner resource. The A operand is used to assign the storage capacity.

4.13 *ANOVA*

The ANOVA command calculates confidence intervals and performs analysis of variance on values stored in the results database (a DOS file). For example,

ANOVA A,B,C

Where the A operand is the file name that holds the results of a simulation. The B operand specifies the values on which the analysis is performed. The C operand specifies the treatment levels.

ANOVA command must be invoked from the data window.

5. *HOW GPSS/PC ™* WORKS

The most basic GPSS model is formed with a GENERATE block and a TERMINATE block, as shown in Figure 3(a). Of course, this simulation does not accomplish any task, since we generate transactions in the first block, and terminate the transactions in the second block. But still in this case, some control needs to be imposed to the simulation in order run it.

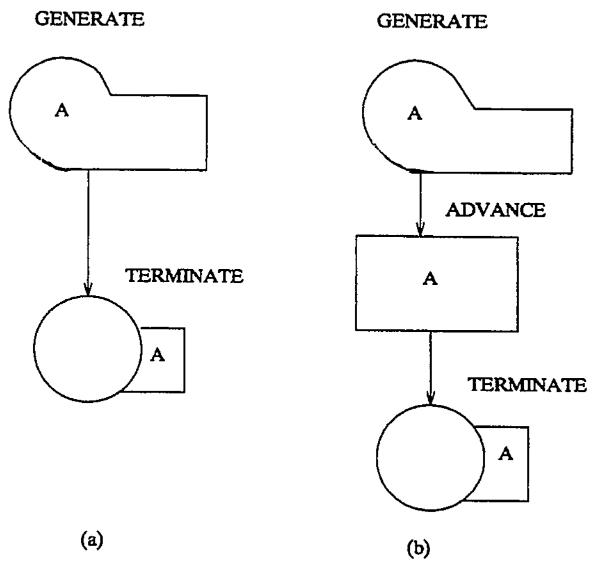


Figure 3- Basic GPSS Models

In the simulation of a GPSS model, the system prepares a **termination counter**, which counts the transactions that will flow through the model blocks. This **termination counter** is transparent to the user, but nevertheless the user must provide the value to store in it. This task is accomplished with the GPSS statement

Start XX

where XX stands for the number of transactions to be terminated for the simulation process. Therefore the GPSS program for the basic model of Figure 3(a) looks like this

```
GENERATE     100
TERMINATE    2
START        1000
```

The previous program will generate a transaction every 100 units of time, and will generate a total of 500 transactions before the simulation stops running. As each transaction passes through the TERMINATE block, it decrements the start count by 2. Therefore, the total time used for the simulation is

Relative Time= 100x500= 50,000 *units*

Let's take under consideration the GPSS block model indicated in Figure 3(b). In this case we have three blocks: GENERATE, ADVANCE, and TERMINATE. ADVANCE is the block which represent the service received by the transaction. In this case we can have the GPSS model given by

```
10       GENERATE     100
20       ADVANCE      120
30       TERMINATE    2
40       END
START    1000
```

In this case, the GPSS output reports give the following results

LINE	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT
10	1	GENERATE	502	0
20	2	ADVANCE	502	2
30	3	TERMINATE	500	0

Let's take into consideration the following model

```
10      GENERATE    100
20      QUEUE       PROS
30      SEIZE       MACH
40      DEPART      PROS
50      ADVANCE     200
60      RELEASE     MACH
70      TERMINATE   2
80      END
START   1000
```

The results are:

LINE	LOC	BLOCK TYPE	ENTRY COUNT	CURRENT COUNT
10	1	GENERATE	1001	0
20	2	QUEUE	1001	500
30	3	SEIZE	501	0
40	4	DEPART	500	0
50	5	ADVANCE	500	0
60	6	RELEASE	500	0
70	7	TERMINATE	500	0

6. *APPLICATION*

For our application in using GPSS/PC™ we consider the local computer network shown in Figure 4.

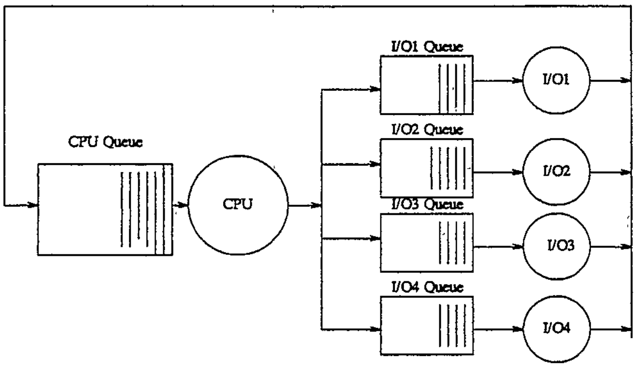


Figure 4- Local Computer Network

The service times are

$$\tau_0=3.20$$
$$\tau_1=3.85$$
$$\tau_2=4.21$$
$$\tau_3=3.26$$
$$\tau_4=3.72$$

Let us assume that we want to calculate the transfer probabilities for full capacity. In our case we have

$$n_1\tau_1=n_2\tau_2=n_3\tau_3=n_4\tau_4$$

Assuming $n_1=1$,

$$n_2=\frac{n_1\tau_1}{\tau_2}=\frac{3.85}{4.21}=0.9144$$

$$n_3=\frac{n_1\tau_1}{\tau_3}=\frac{3.85}{3.26}=1.1809$$

$$n_4=\frac{n_1\tau_1}{\tau_4}=\frac{3.85}{3.72}=1.0349$$

The total average number of processing transactions is

$$\Delta=n_1+n_2+n_3+n_4=1+0.9144+1.1809+1.0349=4.1302$$

The transfer probabilities are

$$p_{01}=\frac{n_1}{\Delta}=\frac{1}{4.1302}=0.2421$$

$$p_{02}=\frac{n_2}{\Delta}=\frac{0.9144}{4.1302}=0.2213$$

$$p_{03}=\frac{n_3}{\Delta}=\frac{1.1809}{4.1302}=0.2859$$

$$p_{04}=\frac{n_4}{\Delta}=\frac{1.0349}{4.1302}=0.2505$$

Before writing the GPSS block diagram we are going to calculate the normalized transfer probabilities that will be needed in order to use the TRANSFER blocks. These probabilities are:

$$p'_{02}=0.2920$$
$$p'_{03}=0.5329$$

The GPSS program is indicated below

```
 ;  GPSS/PC Program File LOCONE.GPS.  (V 2, # 36875)  04-02-1984 02:36:54
10 ****************************************************************
11 *                                                              *
12 *                                                              *
13 *.         WINTER SIMULATION CONFERENCE 1986                   *
14 *              COMPUTER LOCAL NETWORK                          *
15 *               BY RICARDO F. GARZIA                           *
16 *               GPSS/PC- VERSION 2                             *
17 *                                                              *
18 *                                                              *
19 ****************************************************************
20          GENERATE    200,50            ;transactions generation
25          QUEUE       CPUQ              ;cpu queue
30          SEIZE       CPU               ;seize the cpu for processing
35          DEPART      CPUQ              ;depart the cpu queue
40          ADVANCE     320               ;receiving service at the cpu
45          RELEASE     CPU               ;release the cpu
50          TRANSFER    .2421,IOP,IO1     ;transfer to i/o1, p=0.2421
55 IO1      QUEUE       IO1Q              ;i/o1 queue
60          SEIZE       IO1               ;take i/o1 printer
65          DEPART      IO1Q              ;depart the i/o1 queue
70          ADVANCE     385               ;printing time
75          RELEASE     IO1               ;release printer 1
80          TERMINATE   1                 ;terminate the transaction
85 IOP      TRANSFER    .292,IOPP,IO2     ;transfer to i/o2, p=0.2920
90 IO2      QUEUE       IO2Q              ;i/o2 queue
95          SEIZE       IO2               ;take i/o2 printer
100         DEPART      IO2Q              ;depart the i/o2 queue
105         ADVANCE     421               ;printing time
110         RELEASE     IO2               ;release printer 2
115         TERMINATE   1                 ;terminate the transaction
120 IOPP    TRANSFER    .5329,IO4,IO3     ;transfer to i/o3, p=0.5329
125 IO3     QUEUE       IO3Q              ;i/o3 queue
130         SEIZE       IO3               ;take i/o3 printer
135         DEPART      IO3Q              ;depart the i/o3 queue
140         ADVANCE     326               ;printing time
145         RELEASE     IO3               ;release printer 3
150         TERMINATE   1                 ;terminate the transaction
155 IO4     QUEUE       IO4Q              ;i/o4 queue
160         SEIZE       IO4               ;take i/o4 printer
165         DEPART      IO4Q              ;depart the i/o4 queue
170         ADVANCE     372               ;printing time
175         RELEASE     IO4               ;release printer 4
190         TERMINATE   1                 ;terminate the transaction
```

The results of the simulation are:

```
GPSS/PC Report file REPORT.GPS.  (V 2, # 36875)  04-02-1984 02:35:51   page 1

     START_TIME     END_TIME  BLOCKS    FACILITIES  STORAGES    FREE_MEMORY
         0           320612     33           5          0          265680

LINE        LOC          BLOCK_TYPE       ENTRY_COUNT   CURRENT_COUNT   RETRY
 20          1            GENERATE            1612             0           0
 25          2            QUEUE               1612           610           0
 30          3            SEIZE               1002             0           0
 35          4            DEPART              1002             0           0
 40          5            ADVANCE             1002             1           0
 45          6            RELEASE             1001             0           0
 50          7            TRANSFER            1001             0           0
 55         IO1           QUEUE                243             0           0
 60          9            SEIZE                243             0           0
 65         10            DEPART               243             0           0
 70         11            ADVANCE              243             1           0
 75         12            RELEASE              242             0           0
 80         13            TERMINATE            242             0           0
 85         IOP           TRANSFER             758             0           0
 90         IO2           QUEUE                207             0           0
 95         16            SEIZE                207             0           0
100         17            DEPART               207             0           0
105         18            ADVANCE              207             0           0
110         19            RELEASE              207             0           0
115         20            TERMINATE            207             0           0
120         IOPP          TRANSFER             551             0           0
125         IO3           QUEUE                300             0           0
130         23            SEIZE                300             0           0
135         24            DEPART               300             0           0
140         25            ADVANCE              300             0           0
145         26            RELEASE              300             0           0
150         27            TERMINATE            300             0           0
155         IO4           QUEUE                251             0           0
160         29            SEIZE                251             0           0
165         30            DEPART               251             0           0
170         31            ADVANCE              251             0           0
175         32            RELEASE              251             0           0
180         33            TERMINATE            251             0           0

FACILITY      ENTRIES  UTIL.    AVE._TIME AVAILABLE  OWNER PEND INTER RETRY DELAY
  8             243   0.290      383.63       1       1001   0     0     0     0
 15             207   0.271      421.00       1          0   0     0     0     0
 22             300   0.305      326.00       1          0   0     0     0     0
 28             251   0.291      372.00       1          0   0     0     0     0
 CPU           1002   0.999      319.73       1       1002   0     0     0   610

QUEUE          MAX  CONT. ENTRIES ENTRIES(0) AVE.CONT. AVE.TIME  AVE.(-0) RETRY
 CPUQ          610   610    1612          1    306.67  60992.89  61030.75    0
 IO1Q            1     0     243        192      0.01     18.19     96.67    0
 GPSS/PC Report file REPORT.GPS.  (V 2, # 36875)  04-02-1984 02:35:51   page 2
QUEUE          MAX  CONT. ENTRIES ENTRIES(0) AVE.CONT. AVE.TIME  AVE.(-0) RETRY
 IO2Q            1     0     207        158      0.02     30.74    129.86    0
 IO3Q            1     0     300        213      0.00      2.86      9.86    0
 IO4Q            1     0     251        186      0.01     16.99     65.60    0

XACT_GROUP         GROUP_SIZE        RETRY
 POSITION               0              0
```

The analysis of these results allow us to observe the following facts:

1. The total transactions generated is 1612, but at the time of complete the simulation we have 610 transactions in the CPU queue, one transaction receiving service at the CPU, and one transaction receiving service at the printer 1. The printers 2,3 and 4 are empties.

2. The sum of all the terminations counts is

   $$242+207+300+251= 1000$$

   as we suppose to have, since we run the simulation with

   START 1000

3. The transfer probabilities are:

   $$p_{01}=\frac{242}{1000}=0.242$$

   $$p_{02}=\frac{207}{1000}=0.207$$

   $$p_{03}=\frac{300}{1000}=0.300$$

   $$p_{04}=\frac{251}{1000}=0.251$$

   which are very close to the transfer-probabilities stated in the problem.

4. We also observe, that only one transaction reach the CPU without delay, the rest has been delayed in the CPU queue. The average time of the delay in the CPU queue is 18.19.

7. CONCLUSIONS

Although very briefly, in this tutorial, the most important and salient features of GPSS/PC ™ have been presented, which makes this language a very attractive one for the simulation of discrete event models. Some of the features are summarized as follows:

1. **On-line edit mode-** This feature allows the development of a model which carries not syntax errors.
2. **Interruption Capability-** The capability of interrupting the simulation at any time, and the continuation of it is easy to accomplish. During the simulation interruption, model changes can be perform.
3. **Graphic Windows-** During simulation, the capability of changing the graphic windows allows a complete understanding of the simulation under way.
4. **Animation Window-** The animation window, permits to see the animation of the process under simulation as needed.
5. **Simulation Duration-** An important feature of this simulation language is that the time of the simulation is not part of the model definition.

8. REFERENCES

1. Gordon, G.-"A General Purpose Systems Simulation Program"- Proc. EJCC, Washington, DC, New York, Macmillan Publishing Co., Inc., 1961.
2. Gordon, G.-System Simulation- Prentice-Hall Inc., Second Edition, 1978.
3. GPSS/PC ^{TM} Reference Manual- Minuteman Software, 1986.
4. GPSS/PC ^{TM} Tutorials- Minuteman Software, 1986.
5. IBM Corporation-General Purpose Simulation System V, User's Manual- (Form SH 20-0851), White Plains, N.Y.